Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2020

Saku Mikola

TIETOTURVAPOIKKEAMIEN HAVAINNOINTI ELASTALERT-TYÖKALULLA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 39 sivua

Saku Mikola

TIETOTURVAPOIKKEAMIEN HAVAINNOINTI ELASTALERT-TYÖKALULLA

Tietoturvahyökkäyksien taloudellisia ja muita negatiivisia vaikutuksia on mahdollista minimoida sen aikaisella havaitsemisella. Traditionaalisten tietoturvajärjestelmien ja kontrollien pettäessä tietoturvapoikkeaman havainnointi on ainoa keino estää hyökkääjää saavuttamasta tavoitteitaan. Hyökkääjien käyttämien toimintatapojen ja teknologioiden kehittyessä on tärkeää, että tietojärjestelmien puolustajat eivät jää kehityksessä jälkeen.

Opinnäytetyön tarkoituksena oli tutkia ElastAlert-työkalun käyttöä tietoturvapoikkeamien havainnointiin. Opinnäytetyössä tarkasteltiin ElastAlertin toimintaa Elasticsearch-tietokannan sekä muiden Elastic Stackin komponenttien kanssa.

Työssä testattiin ElastAlertin eri toiminnallisuuksia, sekä käsiteltiin ElastAlertista havaittuja puutteita ja mahdollisia ei-haluttuja ominaisuuksia. ElastAlertin eri toiminnalisuuksista ja ominaisuuksista sai työn avulla kattavan kuvan. Työssä luotiin ElastAlertin toiminnan testaamista varten testiympäristö. Työssä testattiin myös password spray -hyökkäyksen havainnoimista Windows-toimialueella. Kootun tiedon avulla ElastAlertin käyttöönottoa harkitseva organisaatio pystyi tekemään päätöksen siitä, että onko ElastAlert heille sopiva työkalu.

ASIASANAT:

havainnointi, lokienhallinta, tietoturva, tietotekniikka

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications technology

2020 | 39 pages

Saku Mikola

DETECTING INFORMATION SECURITY INCIDENTS WITH ELASTALERT

The financial and other negative effects caused by a cyberattacks be minimized by their early detection. When traditional security appliances and controls fail, detecting the incident is the only solution to stop the attackers from reaching their goals. In a time when the methods and technologies used by attackers are constantly evolving, it is important that the defenders of IT-infrastructure come up with up-to-date counter measures.

The objective of the thesis was to review the usage of ElastAlert for detecting and monitoring cybersecurity incidents. In the thesis, the means in which ElastAlert interacts with an Elasticsearch database and the other components of the Elastic Stack are examined.

Different functionalities of ElastAlert were tested in the thesis and the detected shortcomings and possibly unwanted properties of ElastAlert were considered. The thesis provided a comprehensive picture of the different functionalities and properties of ElastAlert. A testing environment was created in the thesis. Detection of a password spraying attack against a Windows domain was also tested. With the information combined in the thesis, an organization which was considering using ElastAlert was able to decide whether ElastAlert was a suitable tool for their needs.

KEYWORDS:

information security, log management, monitoring, information technology

SISÄLTÖ

KUVAT

SANASTO

DC-palvelin	Windows-toimialueen ohjauspalvelin.
DNS-kysely	Kysely, jolla selvitetään verkkotunnuksen Internetin protokollaosoite.
ELK	Elasticsearch, Logstash ja Kibana -ohjelmistojen kokoelma.
IP-osoite	Internetin protokollaosoite, jolla järjestelmät tunnistautuvat tietoverkossa.
JSON	JavaScript Object Notation. Standardisoitu tiedostomuoto.
SSL	Secure Sockets Layer. Tietoverkkosalausprotokolla.
SQL	Structured Query Language. Relaatiotietokantojen kyselykieli.

1 JOHDANTO

Tietoturvan tärkeys ei ole koskaan ollut suurempi. Lähes kaikki ihmiskunnalle tärkeät palvelut ovat siirtyneet erilaisiin tietojärjestelmiin. Ihmiset, valtiot ja yksityiset organisaatiot ovat riippuvaisia näiden ja omien tietojärjestelmiensä toiminnasta. Verkkorikollisuuden, kyberterrorismin ja tietoverkkovakoilun lisääntyessä on selvää, että näitä järjestelmiä pitää suojata. Samanaikaisesti kuitenkin tiedetään, että tietoturvajärjestelmät, joiden tarkoituksena on pelkästään estää tunkeutujia pääsemästä esimerkiksi yrityksen sisäiseen tietoverkkoon, eivät ole täydellisiä. Tämän kaltaisten tietoturvajärjestelmien pettäessä on usein kuitenkin vielä mahdollista estää hyökkääjää saavuttamasta tavoitteitaan. Tietoturvahyökkäyksen taloudellisia ja muita negatiivisia vaikutuksia on mahdollista minimoida aikaisella havaitsemisella. Tietoturvahyökkäyksen elinkaari koostuu useasta eri vaiheesta ennen varsinaista hyökkääjän tavoitteiden saavuttamista. Usein riittää, että hyökkäys estetään jossain hyökkääjän tavoitteita edeltävässä vaiheessa.

Opinnäytetyön aiheena oli tutkia avoimeen lähdekoodiin pohjautuvan ElastAlert-työkalun käyttämistä tietoturvapoikkeamien havainnointiin. Opinnäytetyössä tarkastellaan ElastAlert-työkalun toimintaa Elasticsearch-tietokannan ja muiden Elastic Stackin komponenttien kanssa sekä arvioidaan ElastAlertin eri ominaisuuksia ja mahdollisia puutteita. Havaittuille puutteille esitetään myös vaihtoehtoisia ratkaisuja.

ElastAlert-työkalusta mielenkiintoisen tekee sen käytön potentiaaliset mahdollisuudet. ElastAlertin toiminta eroaa merkittävästi traditionaalisemmista tunkeilijan havaitsemisjärjestelmistä. ElastAlertille on mahdollista kehittää tietoturvapoikkeamien havainnointiin sääntöjä, joissa huomioon otetaan aikaisempia Elasticsearchiin tallennettuja tapahtumia. Traditionaaliset tunkeilijan havaitsemisjärjestelmät, kuten Snort analysoivat tapahtumia reaaliajassa ilman aikaisemmin havaittujen ja tallennettujen tapahtumien kontekstia.

Opinnäytetyön luvussa 2 tarkastellaan tietoturvapoikkeamien havainnointia yleisellä tasolla. Luvussa 3 käsitellään Elastic Stackin eri komponenttien toimintaa. Luvussa 4 tutkitaan ElastAlert-työkalun valmiita toiminnallisuuksia sekä niiden asettamia vaatimuksia. Työn luvussa 5 toteutetaan testiympäristö, jossa ElastAlertin ominaisuuksia testataan sekä kehitetään ratkaisuja testivaiheessa havaittuihin puutteisiin. Viimeisessä luvussa pohditaan opinnäytetyön tavoitteiden saavuttamista, järjestelmän toimintaa sekä jatkokehitystä.

2 TIETOTURVAPOIKKEAMIEN HAVAINNOINTI

2.1 Tietoturvapoikkeamat

Tietoturvajärjestelmillä pyritään takaamaan tietojärjestelmien ja tiedon luottamuksellisuus, eheys ja saatavuus. Luottamuksellisuudella tarkoitetaan salaisen tai yksityisen datan salassapitoa. Tiedon eheydellä tarkoitetaan sitä, että tietoa ei ole voitu luvattomasti muuttaa. Saatavuudella tarkoitetaan yksinkertaisesti, että tietojärjestelmät ovat niille tarkoitettujen henkilöiden ja järjestelmien saatavilla. (Walkowski, D. 2019.) Tietoturvapoikkeamien havainnoinnilla yritetään havaita tietojärjestelmissä tapahtuneita tai tapahtumassa olevia tietoturvahäiriöitä. Tietoturvapoikkeamalla tarkoitetaan tapahtumaa, jossa vähintään yhtä aikaisemmin mainitusta kolmesta tietoturvan peruskäsitteistä rikotaan.

Tietoturvajärjestelmät, kuten esimerkiksi palomuurit ja virustorjuntaohjelmistot estävät hyökkäyksiä eri onnistumisprosentein. Nämä järjestelmät eivät ole siis täydellisiä. On mahdollista, että hyökkääjä pystyy ohittamaan vastaavanlaiset estävät turvajärjestelmät. Kun estävät turvajärjestelmät pettävät on tärkeää, että on olemassa järjestelmiä ja prosesseja, joilla tunkeilija pystytään havaitsemaan organisaation tietojärjestelmissä. (Bejtlich 2013, 4-10.)

Täysin turvallista tietojärjestelmää on käytännössä mahdotonta tehdä. Tietoturvapoikkamien havainnoinnin näkökulmasta onkin siis tärkeää toimia sillä olettamuksella, että hyökkääjät ovat jo tunkeutuneet organisaation tietojärjestelmiin. Puolustajien tehtävä on vaikea. Puolustajat yrittävät estää kaikki mahdolliset luvattomat keinot päästä organisaation tietojärjestelmiin, kun taas hyökkääjien pitää löytää huonoimmassa tapauksessa vain yksi. Tietoturvapoikkeamia monitoroitaessa puolustajilla on kuitenkin eräänlainen etu. Usein riittää, että puolustajat havaitsevat tietoturvahyökkäyksen missä tahansa sen monista vaiheista. Jos hyökkääjä ei ole vielä ehtinyt saavuttamaan tavoitteitaan on puolustajilla mahdollisuus estää tietoturvahyökkäyksen vakavimmat seuraamukset. (Bejtlich 2013, 4-10.)

2.2 Tietoturvahyökkäyksen kulku

Kehittyneet tietoturvahyökkäykset saattavat kestää jopa kuukausia tai vuosia (Bejtlich 2013, 4-8). Onnistunut tietoturvahyökkäys koostuu useasta eri vaiheesta.

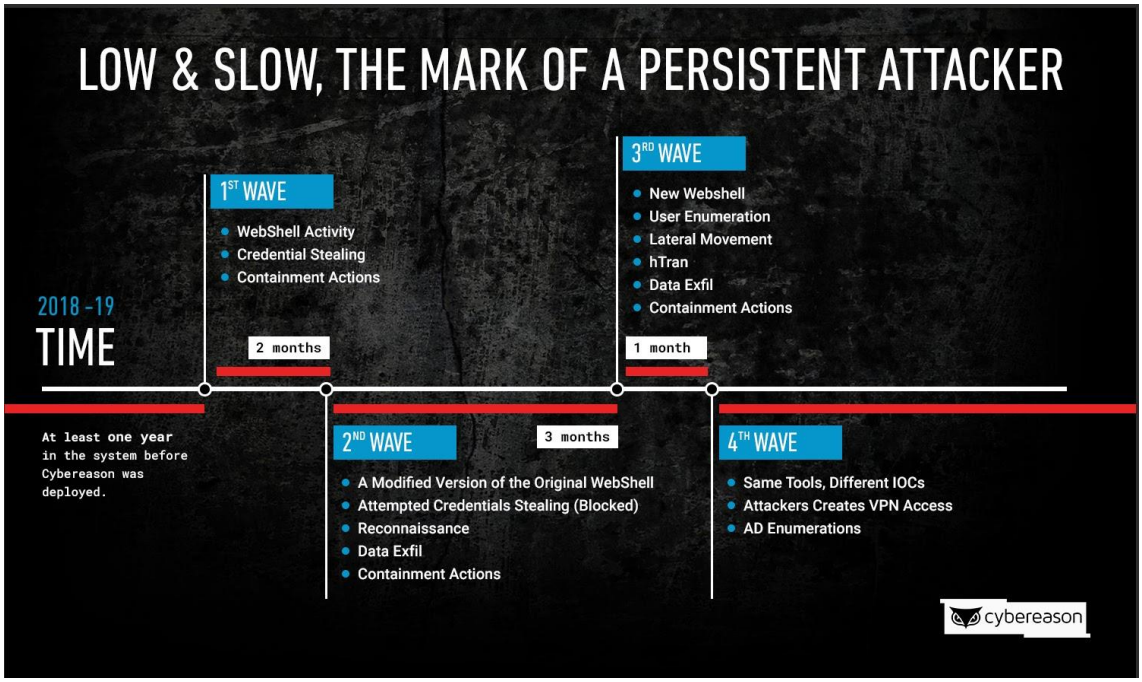
Tietoturvahyökkäyksien kulusta on olemassa useita erilaisia viitekehyksiä. Useat olemassa olevat viitekehykset noudattavat kohtalaisen jäykkää tapahtumien kulkua. Kuvassa 1 on havainnoillistettuna Lockheed Martinin Cyber Kill Chain -viitekehys. Kyseinen viitekehys ei kuitenkaan aina kuvaa todellista tietoturvahyökkäystä erityisen hyvin. Kyseisen viitekehyksen viimeinen vaihe voi pitää sisällään todellisuudessa useita erilaisia vaiheita. Viitekehyksestä voisi helposti saada myös vaikutelman, että jokainen näistä vaiheista suoritetaan vain kerran. Kuitenkin oikeassa tietoturvahyökkäyksessä näitä vaiheita saatetaan toteuttaa monta eri kertaa.

Kuvitellaan esimerkiksi tilanne, jossa hyökkääjän tavoitteena on varastaa yrityksen tuotekehitykseen liittyvää tietoa. Hyökkääjä murtautuu yrityksen sisäisiin tietojärjestelmiin lähettämällä kalasteluviestin yhdelle yrityksen työntekijöistä. Sähköpostin avulla lähetetty kalasteluviesti sisältää haittaohjelman, jonka työntekijä tietämättään asentaa. Työasemalle asennetun haittaohjelman avulla hyökkääjä saavuttaa pysyvän jalansijan yrityksen sisäverkkoon. On hyvin todennäköistä, että saastutettu työasema ei sisällä kaikkea hyökkääjien tavoittelemasta tiedosta. Tähän mennessä Cyber Kill Chain -viitekehys on pitänyt hyvin paikkansa. Seuraavaksi hyökkääjien täytyy alkaa liikkumaan sivusuuntaisesti kuvitteellisen yrityksen tietojärjestelmissä tavoitteidensa saavuttamiseksi. Viitekehyksessä oli kuitenkin jäljellä enää vain yksi vaihe, vaikka hyökkäys oli selkeästi vielä kesken. Hyökkääjät voivat siis suorittaa yrityksen tietojärjestelmien sisällä vielä useita erilaisia vaiheita, joiden järjestystä tai lukumäärää ei voida mitenkään ennalta määrittää. Mitren ATT&CK-viitekehys vastaa tämän kaltaisissa hyökkäyksissä paljon paremmin todellisuutta. Viitekehys ei anna tietoturvahyökkäyksille mitään pakollisia vaiheita tai ennalta määritettyä järjestystä. ATT&CK-viitekehyksessä listataan erilaisia tekniikoita, joita hyökkääjät voivat käyttää sen eri vaiheissa. Esimerkiksi sivusuuntaisen liikkeen alle on listattu Remote Desktop Protocol, jota hyökkääjät voivat käyttää hyödykseen sivusuuntaisessa liikkeessä eri järjestelmien välillä. (Mitre 2020.)

On kuitenkin totta, että usein erityisesti kohdennettujen hyökkäyksien ensimmäinen vaihe on kohteen tiedustelu. Kohteen tiedustelu voi olla Internettiin näkyvissä olevien porttien skannausta tai esimerkiksi sosiaalisen median kautta suoritettavaa tiedon etsintää. Esimerkiksi LinkedIn-sivustolta voi etsiä kohteena olevan yrityksen työntekijöitä ja päätellä heidän profiileistaan mitä ohjelmistoja sekä järjestelmiä kohdeyrityksellä on käytössä. Hyökkäyksen havainnointi tässä vaiheessa on hankalaa. Internetistä saatavilla olevat palvelut ovat aina erilaisten skannaus- ja hyökkäysyritysten kohteena. Kohdennetun uhan havaitseminen muun kohinan seasta onkin siis vaikeaa.

Kuva 1. Lockheed Martinin Cyber Kill Chain -viitekehys (Lockheed Martin N.d).

Tarkastellaan kuvitteellisen hyökkäyksen sijasta oikeasti tapahtunutta tietoturvahyökkäystä. Hyökkäyksen havaitsi yhdysvaltalainen yritys Cybereason, joka julkaisi myös hyökkäyksestä niin sanotun incident response -raportin. Hyökkäyksen kohteena olivat useat kansainväliset teleoperaattorit. Hyökkäys havaittiin vuonna 2018, jonka jälkeen Cybereason alkoi avustamaan uusien hyökkäyksien torjumisessa. Hyökkääjien tavoitteena oli erityisesti varastaa teletunnistetietoja. Hyökkäyksien kulku on kuvattu korkealla tasolla kuvassa 2. Kuvasta selviää, että varsinainen hyökkäyskampanja piti sisällään monia hyökkäyksiä, joista jokainen sisälsi monia eri vaiheita. Hyökkäykset kestivät jopa kuukausia. Kampanjan jokainen hyökkäys alkoi www-palvelimista löytyneiden haavoittuvuuksien hyväksikäytöllä. Tämän jälkeen hyökkääjät alkoivat suorittamaan tiedustelua palvelimesta löytyvistä käyttäjistä ja avoimista porteista. Lopulta hyökkääjät alkoivat leviämään yrityksen tietojärjestelmien välillä saaden haltuunsa useita eri palvelimia ja käyttäjätunnuksia. (Cybereason 2019.)



Kuva 2. Operation Soft Cell -hyökkäyskampanjan kulku (Cybereason 2019).

Raportista voidaankin siis päätellä, että yrityksen estävät tietoturvajärjestelmät epäonnistuivat tehtävässään. Oli siis tärkeää monitoroida yrityksen tietojärjestelmiä tietoturvapoikkeamien varalta. Hyökkäyksien aikajana havainnollistaa hyvin puolustajien saaman edun hyökkääjiin nähden. Hyökkääjät joutuvat suorittamaan yrityksen sisäisissä tietojärjestelmissä useita erilasia vaiheita käyttäen erilasia helposti ja vaikeasti huomattavia tekniikoita. Lisävahingot saatiin estettyä, kun hyökkäys havaittiin ensimmäisen kerran, jonka seurauksena voitiin aloittaa toimenpiteet hyökkääjien torjumiseksi ja tietojärjestelmien turvaamiseksi.

Yritysten tulisikin suorittaa tietoturvapoikkeamien havainnointia usealla eri tavalla. Hyökkäyksiä voidaan havaita esimerkiksi palvelimella suoritetun komennon tai sisäverkossa tapahtuvan epäilyttävän verkkoliikenteen vuoksi. Hyvin suojatussa ja valvotussa yrityksessä verkkoliikennettä tarkkaillaan monesta eri solmukohdasta ja monen eri verkkolaitteen, palvelimen, työaseman ja muiden järjestelmien tuottamat lokit analysoidaan tietoturvapoikkeamien varalta.

3 ELASTIC STACK -KOKOELMA

Elastic Stack koostuu kolmesta avoimeen lähdekoodin pohjautuvasta ohjelmistosta, Elasticsearchista, Logstashista, Kibanasta sekä lisäksi Beats-agenteista. Elastic Stackilla on monta erilaista mahdollista käyttötapausta. Sitä käytetään työkaluna niin luottokorttipetoksien havaitsemisessa, kuin myös erilaisissa koneoppimishankkeissa. (Elastic n.d.a.) Tässä opinnäytetyössä keskitytään kuitenkin Elastic Stackin käyttämiseen ElastAlertin kanssa. Elastic Stackin toiminnan ymmärtäminen oli oleellista opinnäytetyön tekemiselle. ElastAlert-työkalun toiminnan ymmärtäminen osoittautuu todella hankalaksi, jos ei tiedä miten Elastic Stack itsessään toimii. Opinnäytetyön testausvaiheessa tarvitsi myös rakentaa testiympäristö, jossa osalle palvelimista oli asennettuna Elastic Stackin komponentteja.

3.1 Elasticsearch-tietokanta

Elasticsearch on dokumenttipohjainen tietokanta sekä hakukone. Data tallennetaan Elasticsearchiin kuvan 3 muotoisina JSON-dokumentteina. Traditionaalisemmissa, kuten esimerkiksi relaatiomalliin perustuvissa tietokannoissa, data tallennetaan riveittäin eri sarakkeisiin. (Elastic n.d.b.) Dokumenttipohjaisissa tietokannoissa, dokumentissa itsessään kuvaillaan dokumentin eri kentissä esiintyviä arvoja. (Couchbase n.d.).

Elasticsearch tallentaa saamansa JSON-dokumentit indekseihin. Yksi indeksi on siis kokoelma dokumentteja, jonka kaaviossa määritellään mitä kenttiä indeksi sisältää ja minkä tyyppistä dataa kentät sisältävät. Elasticsearch osaa myös lisätä dokumenteissa havaitsemiaan uusia kenttiä indeksin kaavioon ja tunnistaa kentissä olevien arvojen tyyppejä. (Elastic n.d.b.) Tämän takia on yksinkertaista tallentaa eri järjestelmien lokeja Elasticsearch tietokantaan, vaikka tallennettavien lokien kentät vaihtelisivatkin paljon.

Elasticsearchin hakukone pohjautuu Apache Lucene -hakukonekirjastoon. Elasticsearchille on julkaistu useammalla eri ohjelmointikielellä asiakasohjelmia, joilla pystyy muun muassa hakemaan dokumentteja Elasticsearchista. Asiakasohjelmia on julkaistu kielille, kuten Python ja Java. Dataa pystyy hakemaan käyttämällä Elasticsearchin Query DSL -kyselykieltä sekä SQL-tyylisillä kyselyillä. (Elastic n.d.c.)

Elasticsearch on suunniteltu helposti skaalattavaksi järjestelmäksi. Elasticsearch-klusteriin on yksinkertaista lisätä uusia Elasticsearch-palvelimia. Elasticsearch-klusteri osaa automattisesti jakaa datan klusterin eri palvelimien kesken. Elasticsearch suorittaa kuorman tasausta automaattisesti jakamalla kyselyt klusterin eri palvelimille. (Elastic n.d.d.)

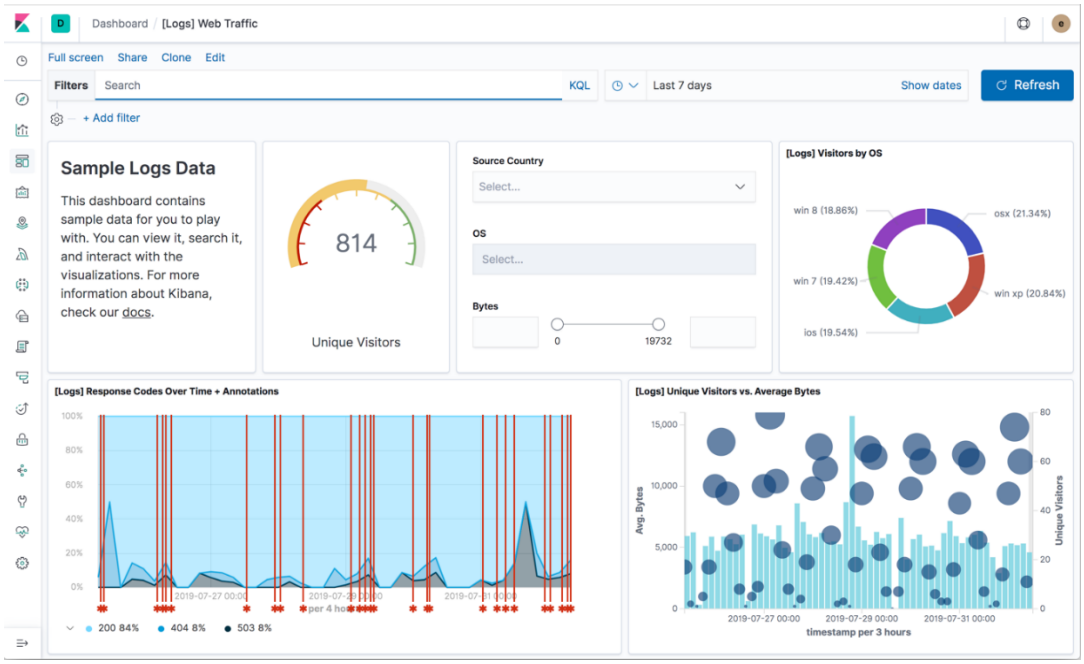
Kuva 3. Esimerkki Elasticsearch-tietokantaan tallennetusta JSON-dokumentista (Elastic 2017).

3.2 Logstash-sovellus

Logstash on avoimeen lähdekoodiin perustuva datan keräykseen ja muuntamiseen tarkoitettu sovellus. Logstashin toiminta perustuu erilaisiin liitännäisiin, joiden avulla dataa voidaan lukea, muuntaa ja lähettää eteenpäin. Logstash voikin siis toimia eräänlaisena kanavana, jonka avulla lokeja voidaan kerätä useasta eri lähteestä. Logstash osaa esimerkiksi jäsentää useita erilaisia lokityyppejä JSON-muotoon ja lähettää eteenpäin Elasticsearchiin. Liitännäisiä datan vastaanottamiseen, lukemiseen ja jäsentämiseen on useita. Esimerkiksi syslog-tyyppisille lokeille ja Elastic Beats-työkaluja varten on luotu valmis liitännäinen. Logstashilla voi myös rikastuttaa luettua dataa esimerkiksi lisäämällä IP-osoitteen maantieteellisen sijainnin lokitapahtumaan ennen sen lähettämistä Elasticsearch-tietokantaan. Logstashin kanssa on myös itse mahdollista rakentaa suodattimia, joilla sille saapuvaa dataa on mahdollista jäsentää JSON-formaattiin. (Elastic n.d.e.)

3.3 Kibana-työkalu

Kibana on Elastic-tuoteperheeseen kuuluva selaimesta käytettävä työkalu. Kibanalla pystyy hallitsemaan Elasticsearchin toimintaa. Kibanan päätarkoitus on kuitenkin auttaa visualisoimaan ja prosessoimaan Elasticsearch-tietokannasta löytyvää dataa. Kibanassa pystyy luomaan erilaisia dynaamisia näkymiä, joiden avulla dokumentteja on helpompi tulkita. Kuvassa 4 on luotu näkymä verkkosivuston vierailuihin liittyvistä dokumenteista. Näkymissä dokumentteja voidaan visualisoida erilaisilla graafeilla. Dokumentteja pystyy myös hakemaan yksilötasolla. Kibanan Discover-työkalun avulla voi esimerkiksi hakea ja suodattaa erilaisia lokitapahtumia halutusta indeksistä. Kibanassa luodaan niin sanottuja indeksikuvioita, joiden avulla dataa voidaan visualisoida ja hakea useammasta indeksistä samanaikaisesti. Kibanassa on vielä lisäksi useita ominaisuuksia, jotka vaativat maksullisen lisenssin. (Elastic n.d.f.)

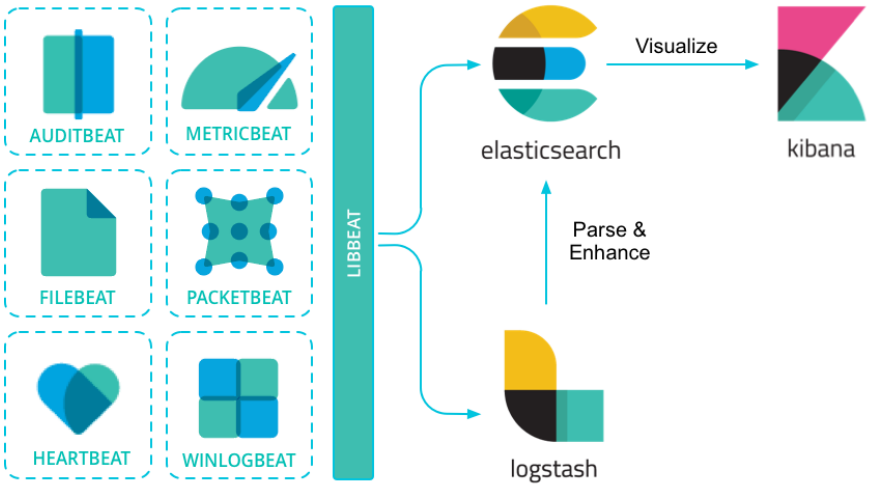


Kuva 4. Kibanan Dashboard-työkalun esimerkki näkymä (Elastic n.d.f).

3.4 Beats-agentit

Beats-sovellukset, ovat esimerkiksi palvelimille asennettuja agentteja, jotka lähettävät dataa Logstashille tai suoraan Elasticsearchiin. Tämän opinnäytetyön testausvaiheessa on käytetty kahta erilaista beats-agenttia. Ensimmäinen niistä on Winlogbeat, jonka

tarkoitus on kerätä Windows-käyttöjärjestelmän tapahtumalokeja. Käytössä on myös ollut Filebeat. Filebeat seuraa paikallisia, sille seurattavaksi määritettyjä lokitiedostoja. Kun Filebeat havaitsee uuden lokirivin seurattavassa tiedostossa, Filebeat lähettää lokin Elasticsearchille tai Logstashille. Muita Elasticin tekemiä beats-agentteja ovat esimerkiksi eri Linux-distribuutioille tarkoitettu Auditbeat tai Packetbeat. Auditbeatin avulla Linux-palvelimen muodostamia auditd-lokeja voidaan lähettää eteenpäin. Packetbeat on suunniteltu keräämään verkkoliikenne-dataa. (Elastic n.d.g.) Kuvassa 5 on havainnollistettu beats-agenttien toimintaa osana Elastic Stack -kokonaisuutta.



Kuva 5. Elastic Stackin toiminta kokonaisuutena (Elastic n.d.g).

4 ELASTALERT-TYÖKALU

ElastAlert on työkalu, jonka avulla pyritään havaitsemaan poikkeamia Elasticsearch-tietokannassa olevasta datasta sekä muodostamaan niistä erilaisia hälytyksiä. Työkalun on kehittänyt yhdysvaltalainen yhtiö Yelp. Yelp aloitti ElastAlertin kehityksen, kun se huomasi tarpeen työkalulle, jonka avulla se voisi automaattisesti muodostaa yksinkertaisia hälytyksiä Elasticsearch-tietokannasta löytyvistä poikkeamista. Yelpin suunnitelma oli luoda ElastAlertista helppokäytteinen, luotettava ja mukautettavissa oleva työkalu. (Yelp 2020a.) Yelp on vuonna 2004 perustettu pörssiyhtiö, jonka omistama yelp.com-verkkosivusto toimii alustana erilaisien paikallisien yrityksien, kuten ravintoloiden, tietojen jakamiselle ja verkkosivuston käyttäjien luomille arvosteluille. Vuonna 2019 Yelpin liikevaihto oli miljardi Yhdysvaltain dollaria. (Yelp 2019.)

ElastAlert-työkalun avulla havaitut poikkeamat voivat esimerkiksi olla tietyn merkkijonon löytäminen dokumentin kentästä tai suodattimiin sopivien dokumenttien äkillinen kasvu tai lasku (Yelp 2020a). ElastAlert-työkalua voidaan siis käyttää osana tietoturvamonitorointi-prosessia. Hunting ELK -projekti on yksi esimerkki, jossa näin on tehty. Hunting ELK -projekti on Microsoftin omistamalla GitHub-verkkosivustolla ylläpidetty avoimeen lähdekoodiin perustuva tietoturvamonitorointiin käytettävä järjestelmä, joka koostuu useasta eri ohjelmistosta. (HELK 2020.) HELK-projekti on kuitenkin vielä alkuvaiheessa, ja projektissa käytetään ainakin vielä hyvin pientä osaa ElastAlertin ominaisuuksista.

ElastAlert pohjautuu avoimeen lähdekoodiin. ElastAlertin lähdekoodi ja dokumentaatio on saatavilla GitHubissa. ElastAlert-projekti on aktiivinen GitHubissa ja edelleen jatkuvassa kehityksessä. Yelpin kehittäjät ovat hyväksyneet projektiin lisäyksiä myös ulkoisilta osapuolilta. Tällä hetkellä ElastAlert-projektin kehitykseen on osallistunut 131 kehittäjää. (Yelp 2020b.) ElastAlert-työkalu on myös ladattavissa Python-ohjelmointikielen pip-kirjastonhallintasovelluksen avulla. Pip-työkalun avulla ladattava versio ei kuitenkaan aina sisällä kaikkia uusimpia lisäyksiä tai muutoksia mitä GitHub-projektiin on tehty. Ensimmäinen pip-työkalulla ladattava versio on julkaistu Python Package Index sivustolle 9. kesäkuuta 2016. (Yelp 2020c.)

4.1 ElastAlert verrattuna muihin järjestelmiin

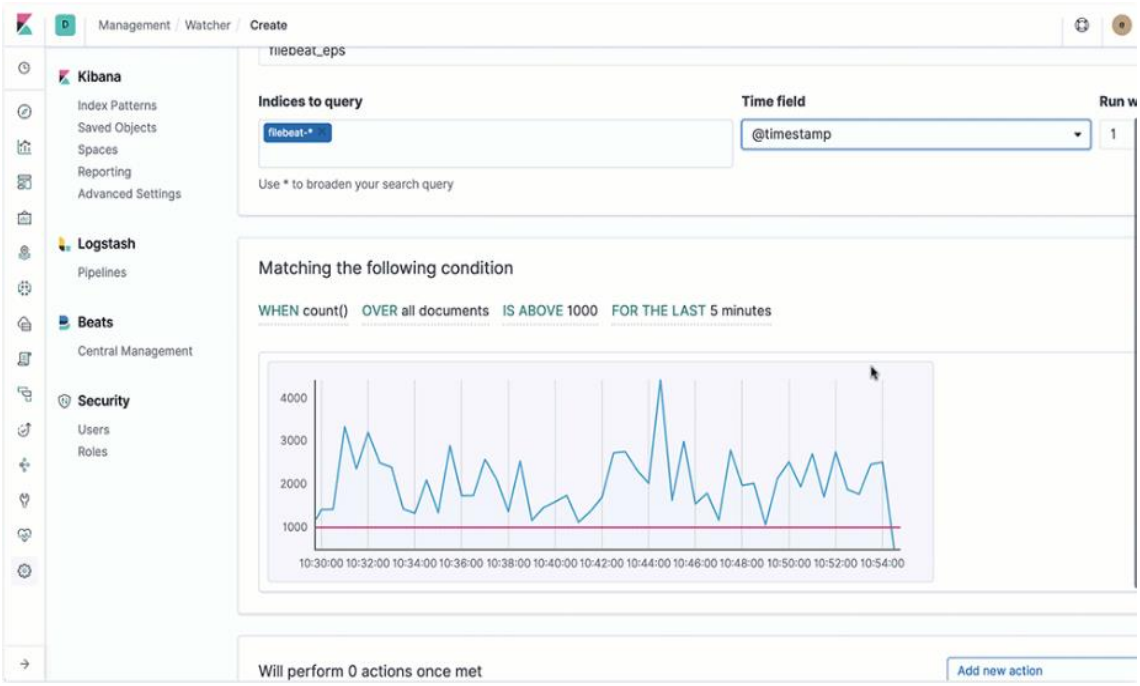
ElastAlert eroaa toiminnaltaan huomattavasti muista poikkeamien havainnointiin tarkoitetuista traditionaalisemmista tunkeilijan havaitsemisjärjestelmistä. ElastAlert-työkalun avulla sääntöpohjaista havainnointia ei suoriteta reaaliajassa. Esimerkiksi verkkoliikennetapahtumat tallennetaan ensin keskitetysti Elasticsearch-tietokantaan, josta dataa sitten haetaan. Tämä toimintamalli erottaa ElastAlertin muista järjestelmistä. Esimerkiksi tunkeilijan havaitsemisjärjestelmä Snort suorittaa sääntöpohjaista analyysia verkkoliikenteestä reaaliajassa. Verkkoliikennettä voidaan esimerkiksi peilata Snortille useasta tietoverkon solmukohdasta, jonka jälkeen Snort luo sille määritettyjen sääntöjen avulla hälytyksiä. (Prakasha S. 2016.) Tällä toimintamallilla on omat hyötynsä ja haittansa. ElastAlertin jälkikäteinen analyysi mahdollistaa suuremman kontekstin käytön. ElastAlertin hälytyksissä voidaan siis ottaa huomioon aikaisemmin tapahtuneita hälytyksiä tai itse aikaisempaa verkkoliikennettä. Tämä mahdollistaa piikkien ja muiden tilastollisien poikkeamien sääntöpohjaisen havainnoinnin. Tulee kuitenkin ottaa huomioon, että suurien datamäärien hakeminen Elasticsearchista voi rasittaa sitä huomattavasti.

ElastAlert ei kuitenkaan korvaa traditionaalisempia tunkeilijan havaitsemisjärjestelmiä. ElastAlertin käyttö niiden rinnalla mahdollistaa uudenlaisten hälytyksien luomisen. On erittäin paljon tapauksia missä ElastAlert on huonompi ratkaisu hälytyksen luomiselle kuin esimerkiksi Snort. Esimerkiksi sääntö, jolla pyritään havaitsemaan DNS-kysely tietylle huonomaineiselle verkkotunnukselle, toimii paljon suoraviivaisemmin Snortin kuin ElastAlertin kanssa. Tämänkaltainen hälytys ei vaadi mitään muuta kontekstia sen ympärille. Tässä tapauksessa on paljon yksinkertaisempaa ja tehokkaampaa luoda hälytys heti kun kyseinen DNS-kysely havaitaan. Tämänkaltaisen säännön lisääminen Snorttiin lisää rasitusta sille hyvin vähän. Jos tämä sääntö toteutettaisiin ElastAlertilla aiheutuisi siitä vain turhaa rasitusta Elasticsearch-tietokannalle. Jokainen ElastAlertille luotu sääntö tulee suunnitella hyvin tarkkaan sillä jo yksi sääntö voi rasittaa Elasticsearchia todella paljon. Snort vertailee vastaanottamaansa verkkoliikennettä omaan säännöstöönsä, kun taas ElastAlert joutuu jokaisen säännön kohdalla tekemään ensin erillisen kyselyn Elasticsearchiin.

Traditionaalisten tunkeilijan havaitsemisjärjestelmien käyttöönotto tietoturvapoikkeamien havainnointiin on yrityksille myös huomattavasti yksinkertaisempaa kuin ElastAlertin, vaikka lokeja tai verkkoliikennettä tallennettaisiinkin valmiiksi jo Elasticsearch-tietokantaan. Esimerkiksi Snort sisältää suuren valmiin säännöstön poikkeamien

havainnointiin (Fellinge J. 2003). ElastAlertin jokainen sääntö pitää tehdä itse sillä tämän kaltaista säännöstöä ei ole mistään saatavilla. Tästä syystä ElastAlert on sopiva työkalu organisaatiolle vain, jos siitä haetaan lisäapua olemassa olevien tietoturvapoikkeamien havainnointiin tarkoitettujen järjestelmien ja prosessien rinnalle.

Elastic Stackin Gold-lisenssissä tarjotaan Elasticin omaa Alerting-työkalua (Elastic n.d.h). Elastic Stackin Alerting-työkalulla pystyy tekemään hyvin samanlaisia sääntöjä sekä hälytyksiä kuin ElastAlert-työkalulla. Alerting-työkalussa etuna on sen yksinkertainen ja selkeä käyttöliittymä sääntöjen ja hälytysten luomiselle ja hallitsemiselle. Kuvassa 6 on Alerting-työkalun Kibanassa toimiva käyttöliittymä. ElastAlert ei sisällä varsinaista käyttöliittymää, jonka kautta sääntöjä ja hälytyksiä voitaisiin hallita. Yksi suuri etu ElastAlertissa on se, että ElastAlertin käyttö Elastic Stackin kanssa ei aiheuta lisenssikustannuksia. ElastAlertin kanssa yrityksellä voi siis olla käytössä Elastic Stackin ilmaisversio. ElastAlertilla on myös Kibana-liitännäinen. Kibana-liitännäisen toiminnallisuus ei kuitenkaan välttämättä anna suurta lisäarvoa verrattuna sääntöjen hallitsemiseen manuaalisesti. ElastAlertin Kibana-liitännäisen kehitys on ollut hidasta. Projektin GitHub sivulla ei ole tapahtunut yhtään päivitystä yhdeksään kuukauteen. Yhteensopivaa liitännäistä uusimalle Elastic Stack 7.6 -versiolle ei ole vielä julkaistu. (Bitsensor 2020.)



Kuva 6. Elastic Stackin Alerting-työkalun käyttöliittymä (Elastic n.d.h).

4.2 Vaatimukset ja toiminta

ElastAlert voidaan asentaa mille tahansa Linux-distribuutiolle. Ainoa varsinainen vaatimus Linux-palvelimelle on, että sille pystyy asentamaan Python-ohjelmointikielen tulkin, jonka versio on 3.6. (Yelp 2020d.) Järjestelmällä tulee kuitenkin olla kohtuullinen määrä keskusmuistia, sillä ElastAlert säilyttää Elasticsearchista ladattujen dokumenttien tietoja järjestelmän keskusmuistissa. Säilytettävän datan määrä riippuu sääntöön palautettujen dokumenttien määrästä sekä siitä kuinka pitkäksi säännön aikaikkuna on määritetty. Tämän toiminnallisuuden avulla ElastAlert tietää mitkä dokumentit se on aikaisemmin nähnyt ja mitkä ovat uusia. (AUTO1 Group 2018.)

ElastAlert vaatii tietysti vielä toimiakseen Elasticsearch-tietokannan, minne kyselyjä tehdään sekä minne ElastAlertin tilaan liittyviä tietoja tallennetaan. ElastAlert kirjoittaa Elasticsearchiin tietoja ajetuista säännöistä, hiljennetyistä säännöistä, ElastAlertin kohtaamista virhetiloista sekä lähetetyistä hälytyksistä. Elasticsearchista haettavien dokumenttien pitää olla aikaleimattu ISO8601- tai Unix-formaatin mukaisesti. (Yelp 2020d.)

Kun ElastAlert käynnistetään, se lukee ensimmäiseksi config.yaml-konfiguraatiotiedostoa. Config.yaml-konfiguraatiotiedostossa määritellään yleisiä asetuksia liittyen ElastAlertin toimintaan. Siellä määritellään esimerkiksi mihin IP-osoitteeseen, porttiin sekä indeksiin ElastAlert kirjoittaa tilastaan. Samassa tiedostossa määritellään myös mihin polkuun sääntötiedostot tulee tallentaa. ElastAlert suorittaa kaikki löytämänsä säännöt, config.yaml-tiedostossa määritetyllä intervallilla. Myös sääntöjen suorittamien kyselyjen koko asetetaan samassa tiedostossa. (Yelp 2020d.)

ElastAlertin toiminta pohjautuu modulaarisiin komponentteihin: sääntöihin ja hälytyksiin. ElastAlertissa on muitakin modulaarisia osia, mutta niiden käyttö ei ole pakollista, eikä aina edes tarpeellista. (Yelp 2020a.)

4.3 Säännöt

Säännöt määrittelevät mitä dataa Elasticsearch-tietokannasta haetaan ja mitä vastaanotetulle datalle tehdään. Säännöt sijaitsevat ElastAlertin konfiguraatiotiedostossa määritetyssä polussa. Jokaisella säännöllä on oma tiedostonsa. Säännöt kirjoitetaan YAML-formaatin mukaisesti. (Yelp 2020e.)

Jokaisessa säännössä määritellään säännön tyyppi. Säännön tyyppi määrittelee missä tilanteissa hälytys tai hälytykset tehdään. Kaikki sääntötyypit jakavat keskenään yleisiä asetuksia. Jokaisessa säännössä pitää määrittää missä IP-osoitteessa kyseltävä Elasticsearch on, ja mitä porttia se kuuntelee. Säännöissä pitää myös määrittää kyseltävä indeksi tai indeksit sekä käytettävät hälyttimet säännön lauetessa. (Yelp 2020e.)

Säännöillä on paljon myös muita yleisiä asetuksia, jotka toimivat kaikkien sääntötyyppien kanssa. Säännössä voidaan määrittää käyttäjätunnus ja salasana, jota käytetään Elasticsearchia kysellessä. (Yelp 2020e.) Muita asioita mitä lähes kaikissa säännöissä kannattaa ehdottomasti määrittää ovat säännön nimi, kuvaus, suodattimet ja käytetäänkö Elasticsearchin ja ElastAlertin välisessä kommunikaatiossa SSL-salausta. Jos salausta ei käytetä, niin silloin mahdollisesti arkaluontoiset lokitiedot ovat alttiina salakuuntelulle. ElastAlert käyttää Elasticsearchin Query DSL -kyselykielen mukaisia suodattimia kyselyissään (Yelp 2020e). Kuvassa 7 on esimerkki Query DSL -kyselykielen käytöstä sääntötiedoston sisällä.

```
filter:
- query:
    query_string:
      query: "username: bob"
- query:
    query_string:
      query: "_type: login_logs"
- query:
    query_string:
      query: "field: value OR otherfield: othervalue"
- query:
    query_string:
       query: "this: that AND these: those"
```

Kuva 7. Esimerkki query_string-suodatintyypin käytöstä ElastAlert säännössä (Yelp 2020f).

Jokaisella sääntötyypillä on yleisten asetuksien lisäksi, omat vaadittavat ja vapaavalinnaiset asetuksensa. Useissa säännöissä määritetään aikaikkuna, jossa esimerkiksi dokumenttien määrää vertaillaan johonkin kynnysarvoon. ElastAlertin dokumentaatiossa on kohtuu hyvin kerrottu mitä asetuksia eri sääntötyypit vaativat. (Yelp 2020e.) Dokumentaatiota lukiessa tuli kuitenkin heti selväksi, että tiettyjen asetuksien kuvaukset ja toiminta on selitetty hieman puutteellisesti tai epäselvästi. Seuraavissa kappaleissa on selitetty ElastAlertin sisäänrakennetut sääntötyypit.

Any tekee hälytyksen jokaisesta Elasticsearchista palautetusta dokumentista. Haettavia dokumentteja voidaan kuitenkin rajata säännössä määritettyjen suodattimien avulla. (Yelp 2020e.)

Blacklist tekee hälytyksen, jos palautetun dokumentin valitussa kentässä sopimattomaksi määritetty arvo. Sopimattomat arvot voidaan määrittää sääntötiedostossa tai listata erillisessä tiedostossa. (Yelp 2020e.)

Whitelist tekee hälytyksen, jos dokumentin valittu kenttä ei sisällä ennalta hyväksyttyä arvoa. Toimii käytännössä päinvastaisesti blacklist-sääntötyyppiin verrattuna. (Yelp 2020e.)

Change tekee hälytyksen, jos dokumentin valitun kentän arvo muuttuu, verrattuna aikaisempaan dokumenttiin tietyn aikaikkunan sisällä (Yelp 2020e).

Frequency tekee hälytyksen, jos tietyn aikaikkunan sisällä on enemmän kuin ennalta määritetty määrä dokumentteja (Yelp 2020e).

Spike vertailee kahden eri aikaikkunan dokumenttien määrää keskenään. Hälytys muodostetaan, kun nykyisen aikaikkunan dokumenttien määrä ylittää verrattavan aikaikkunan dokumenttien määrän tarpeeksi moninkertaisesti. (Yelp 2020e.)

Flatline tekee hälytyksen, jos tietyn aikaikkunan sisällä tulee vähemmän kuin ennalta määritetty määrä dokumentteja (Yelp 2020e).

New_term tekee hälytyksen, jos jossain seurattavista kentistä on sellainen arvo, jota ElastAlert ei ole ennen nähnyt. ElastAlertin käynnistyessä se kyselee Elasticsearchista dokumentteja ja aggregoi tuloksista seurattavien kenttien tunnettujen arvojen listan. (Yelp 2020e.)

Cardinality tekee hälytyksen, sen perusteella kuinka monta uniikkia arvoa valittu kenttä sisältää määritetyn aikaikkunan sisällä. Hälytys voidaan tehdä, kun asetettu kynnysarvo ylitetään tai alitetaan. (Yelp 2020e.)

Metric aggregation -säännössä määritetään kenttä, jonka arvot syötetään valittuun funktioon. Mahdollisia funktioita ovat maksimin tai minimin löytäminen, lukujen summa, lukujen keskiarvo, kardinaliteetti ja arvojen määrän laskeminen. Tämän jälkeen tarkistetaan, onko funktion tuottama luku säännön yläkynnysarvon ja alakynnysarvon välissä. (Yelp 2020e.)

Spike aggregation -sääntötyyppi on käytännössä metric aggregation ja spike-sääntötyyppien yhdistelmä. Sen sijaan, että vertailtaisiin valitun funktion tuottamaa lukua ennalta määritettyihin kynnysarvoihin, sitä vertaillaan edeltävän aikaikkunan vastaavaan laskutoimitukseen. Jos arvot eroavat tarpeeksi moninkertaisesti, tehdään hälytys. (Yelp 2020e.)

Percentage match tekee hälytyksen, jos palautettujen dokumenttien joukossa on liikaa tai liian vähän haluttua dokumenttityyppiä. Säännössä määritetään minkä prosentin ylittyessä tai alittuessa hälytys tehdään. (Yelp 2020e.)

Olemassa olevien sääntötyyppien lisäksi pystyy tekemään myös omia sääntötyyppejä. ElastAlertin dokumentaatiossa näytetään esimerkki miten oman sääntötyypin voi luoda. Esimerkissä luodaan sääntötyyppi, joka hälyttää, jos geneerinen kirjautumistapahtuma tapahtuu sopimattomaan kellonaikaan. Omat sääntötyypit luodaan Python-ohjelmointikielellä. Kuvassa 8 on viitekehys oman sääntötyypin luomiselle. Omaa sääntötyyppiä luodessa pitää vähintään tehdä luokka, joka perii ElastAlertin tiedostoissa määritetyn RuleType-luokan sekä implementoida oman luokan sisällä add_data-funktio. (Yelp 2020g.) Oman sääntötyypin luominen vaikuttaa erittäin yksinkertaiselta Pythonia osaavalle.

```
class AwesomeNewRule(RuleType):
    # ...
    def add_data(self, data):
        # ...
    def get_match_str(self, match):
        # ...
    def garbage_collect(self, timestamp):
        # ...
```

Kuva 8. Viitekehys oman sääntötyypin luomista varten (Yelp 2020g).

4.4 Hälyttimet

Hälyttimet määrittelevät tehtävän toiminnon, kun sääntö laukeaa. Hälytin siis määrittelee kanavan, mitä kautta säännön laukeamisesta ilmoitetaan. Säännössä voi olla listattuna useampi eri hälytin. Hälyttimen asetukset määritellään sääntötiedostossa. Säännön sisällä määritellään esimerkiksi mihin sähköpostiosoitteisiin hälytys lähetetään sähköpostihälytintä käytettäessä. (Yelp 2020e.)

ElastAlertin mukana tulee useita erilaisia valmiita hälyttimiä. Valmiita hälyttimiä on luotu ohjelmille, kuten Slack, Microsoft Teams, JIRA ja Telegram. Command-hälyttimen avulla pystyy ajamaan halutun komennon palvelimella johon ElastAlert on asennettu. (Yelp 2020e.) Jo pelkästään kuvassa 9 havainnollistetun command-hälytintyypin kanssa ElastAlertin hälytyksien integroiminen yrityksen omiin järjestelmiin on yksinkertaista.

```
alert:
  - command
command: ["/bin/send_alert", "--username", "%(username)s"]
```

Kuva 9. Esimerkki command-hälyttimen käytöstä sääntötiedostossa (Yelp 2020e).

Tarvittaessa ElastAlerttiin pystyy luomaan omia hälyttimiä. Yrityksen omien järjestelmien integrointi ElastAlert-työkalun kanssa onnistuukin vielä elegantimmin luomalla oman hälyttimen kuin käyttämällä command-hälytintä. Itsetehdyn hälyttimen sisällä voi käyttää säännössä määritettyjä kenttiä sekä sääntöön osuneen dokumentin kenttiä. Oma hälytin luodaan Python-ohjelmointikielellä. Kuvassa 10 on viitekehys hälyttimen luomiselle. Hälyttimessä olevan luokan pitää periä Alerter-luokka sekä luokan sisällä pitää implementoida kaksi jäsenfunktiota (Yelp 2020h).

```
class AwesomeNewAlerter(Alerter):
    required_options = set(['some_config_option'])
    def alert(self, matches):
        ...
    def get_info(self):
        ...
```

Kuva 10. Viitekehys oman hälyttimen luomista varten (Yelp 2020h).

4.5 Enhancement-moduulit

Enhancement-moduulit ovat Python-ohjelmointikielellä kirjoitettuja komentosarjoja, joiden avulla sääntöön osuneita dokumentteja voidaan muokata ennen hälytyksen lähettämistä. Enhancement-moduulilla pystyy esimerkiksi perumaan hälytyksen lähetyksen, jos enhancement-moduulin sisällä määritetyt ehdot täyttyvät. Sääntötiedostossa määritellään enhancement-moduuli, jos sellaista halutaan säännön ohella käyttää. (Yelp 2020i.) ElastAlertin mukana ei tule valmiita enhancement-moduuleja.

Hälytykseen pystyy myös lisäämään lisäinformaatiota. ElastAlertin dokumentaatiossa näytetään esimerkki, jossa luodaan enhancement-moduuli, joka lisää sääntöön osuneeseen dokumenttiin linkin. Linkki luodaan dokumentissa olevasta domain-kentästä ja se osoittaa whois-verkkosivustolle, josta nähdään verkkotunnuksen julkisia tietoja. (Yelp 2020i.) Tämänkaltaisten linkkien avulla voidaan esimerkiksi nopeuttaa ja suoraviivaistaa hälytyksestä aiheutuvaa analysointiprosessia.

Enhancement-moduuleja voidaan käyttää monella eri tavalla. Voidaan esimerkiksi luoda moduuli, joka tarkistaa, että dokumentin aikaleima on tietyn aikaikkunan sisällä. Tämän tyylistä enhancement-moduulia voidaan käyttää esimerkiksi any-sääntötyypin kanssa. Esimerkki tämän kaltaisesta säännöstä voisi olla sääntö, joka hakee ja suodattaa kirjautumistapahtumia tärkeälle palvelimelle. Tämän säännön ja enhancement-moduulin avulla olisi silloin mahdollista hälyttää pelkästään yöaikaan tapahtuvista kirjautumisista kuten kuvassa 11 kuvatussa enhancement-moduulissa. (0x29A tech blog 2019.)

hour_range_enhancement.py

```
1  import dateutil.parser
2
3  from elastalert.enhancements import BaseEnhancement
4  from elastalert.enhancements import DropMatchException
5
6  class HourRangeEnhancement(BaseEnhancement):
7      def process(self, match):
8          timestamp = dateutil.parser.parse(match['timestamp']).time()
9                  # UTC time (I'm from france ==> UTC + 2)
10         time_start = dateutil.parser.parse("4:00").time()
11                 time_end = dateutil.parser.parse("20:00").time()
12         if timestamp < time_start or timestamp > time_end:
13             raise DropMatchException()
```

Kuva 11. Enhancement-moduuli, jolla hälytyksiä voidaan estää kellonajan perusteella (0x29A tech blog 2019).

4.6 Rules loader -komentosarjat

Rules loader -komentosarjat kirjoitetaan Python-ohjelmointikielellä. Rules loader -komentosarjojen avulla sääntötiedostoja voidaan säilyttää jossain muualla kuin palvelimella johon ElastAlert on asennettu. ElastAlertin dokumentaatiossa näytetään esimerkki, jossa luodaan rules loader -komentosarja, joka hakee sääntötiedostoja MongoDB-tietokannasta käyttäen Python-ohjelmointikielelle tehtyä pymogo-kirjastoa. (Yelp 2020j.)

Rules loader -komentosarjat mahdollistavat esimerkiksi järjestelmän rakentamisen, missä ElastAlertin säännöstöä hallittaisiin itsetehdyn käyttöliittymän kautta ja säännöstöä säilytettäisiin valitussa tietokannassa. Tällä tavalla ElastAlertin sääntöjen luominen, muokkaaminen ja muu hallinta pystyttäisiin mahdollisesti integroimaan yrityksen jo valmiiksi käytössä oleviin järjestelmiin. Rules loader -komentosarjan teko on kohtuu yksinkertaista. Kuvassa 12 on viitekehys oman komentosarjan luomiselle.

```
class AwesomeNewRulesLoader(RulesLoader):
    def get_names(self, conf, use_rule=None):
        ...
    def get_hashes(self, conf, use_rule=None):
        ...
    def get_yaml(self, rule):
        ...
```

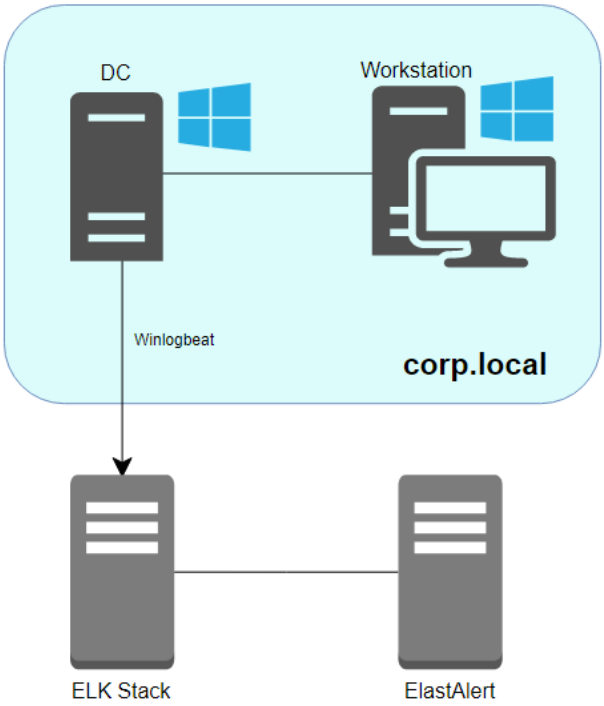
Kuva 12. Viitekehys oman rules loader -komentosarjan luomiselle (Yelp 2020j).

5 ELASTALERT-TYÖKALUN TESTAUS

ElastAlertin testauksen tavoitteena oli testata sen eri toiminnallisuuksia sekä tutkia havaittuja puutteita. Testiympäristön avulla oli myös tehtävänä havainnollistaa esimerkkitilanteita, jossa ElastAlerttia voidaan käyttää tietoturvapoikkeamien havainnointiin.

5.1 Testiympäristö

Opinnäytetyön testivaiheessa käytetty testiympäristö on havainnollistettu kuvassa 13. Testiympäristö on toteutettu virtuaalikoneilla Oraclen VirtualBox-sovelluksessa. ELK- ja ElastAlert-palvelimille on asennettu Debian 10.3 (Buster) -käyttöjärjestelmä. Testiympäristön corp.local-toimialue koostuu toimialueen ohjauspalvelimesta (DC) ja yksittäisestä Windows 10 -työasemasta. Windows toimialueen päätarkoitus testiympäristössä on generoida lokitapahtumia ja lähettää ne Winlogbeat-agentin avulla ELK Stack -palvelimella sijaitsevalle Logstashille. Myös Kibana ja Elasticsearch-tietokanta sijaitsevat ELK-palvelimella. ElastAlert on siis asennettuna erilliselle palvelimelle.



Kuva 13. Testiympäristö

Corp.local-toimialueelle luotiin myös noin 100 testikäyttäjää spike-sääntötyypin testausta varten. Käyttäjien luonti tehtiin DC-palvelimella kuvan 14 mukaisella yksinkertaisella Powershell-komentosarjalla.

```
1..100 | Foreach-Object {
$f = -join ((97..122) | Get-Random -Count 2 | % {[char]$_})
$r = Get-Random -Minimum 1000 -Maximum 9999
$k = $f + $r
$pass = "Password" + $r
New-ADUser -Name "$k" -UserPrincipalName "$k@corp.local" -Path "ou=testou,dc=corp,dc=local"
Set-ADAccountPassword -Identity $k -Reset -Newpassword (ConvertTo-SecureString -AsPlaintext $pass -Force)
Enable-ADAccount -Identity $k}
```

Kuva 14. Powershell-komentosarja Windows-toimialueen testikäyttäjien luomiseen.

Testiympäristössä ElastAlertin ja Elasticsearchin välillä ei käytetty SSL-salausta tai autentikointia. ElastAlert määritettiin kirjoittamaan sääntöihin ja hälytyksiin liittyvän metadatan ELK-palvelimen Elasticsearchiin.

Testivaiheessa ElastAlertista paljastui myös mahdollisesti ei-haluttu ominaisuus. Oletusarvoisena ElastAlert jatkaa esimerkiksi uudelleen käynnistyksen yhteydessä säännön suoritusta sen viime suorituksesta eteenpäin. Voi syntyä siis tilanne, jossa sääntö on ollut hiljennettynä esimerkiksi virhetilan vuoksi hyvin pitkän aikaa ja kun ElastAlert käynnistetään uudelleen se jatkaa säännön suoritusta monen päivän takaa. Tämä on hyvä ominaisuus sen takaamiselle, että hälytyksiä ei jää virhetilan vuoksi tekemättä, mutta suuren lokimäärän kanssa tämä saattaisi aiheuttaa turhan paljon rasitusta Elasticsearch-tietokannalle. Config.yaml-tiedostossa voidaan kuitenkin määrittää ajanjakso mistä sääntöjen suoritusta saa maksimissaan jatkaa. Kuvassa 15 esitetyssä config.yaml-tiedostossa ElastAlert on asetettu jatkamaan vain alle 5 minuuttia vanhoja sääntöjen suoritusta. ElastAlert on myös asetettu uudelleen yrittämään hälytyksen lähettämistä virhetilanteessa kahden vuorokauden ajan.

```
rules_folder: rules

run_every:
  minutes: 1

buffer_time:
  minutes: 5

es_host: 192.168.88.60
es_port: 9200

writeback_index: elastalert_status

old_query_limit:
  minutes: 5

alert_time_limit:
  days: 2
```

Kuva 15. Testiympäristön config.yaml-tiedosto.

5.2 Oman hälyttimen teko

ElastAlert kirjoittaa sääntöjen muodostamista hälytyksistä dokumentin config.yaml-tiedostossa määritettyyn metadataindeksiin (Yelp 2020d). ElastAlertissa ei kuitenkaan ole suoraan toiminnallisuutta, minkä avulla hälytyksiä voitaisiin erotella eri indekseihin kyselyn indeksin perusteella. Hälytyksen indeksin voi itsetehdyn hälyttimen sisällä päätellä kahdella tavalla. Kun sääntö aiheuttaa hälytyksen, se lähettää hälyttimelle sen dokumentin mikä aiheutti säännön laukeamisen. Tätä toimintamallia ei kuitenkaan voida käyttää kaikissa säännöissä.

Flatline-sääntötyypin kanssa sääntö ei varsinaisesti laukea mihinkään dokumenttiin, vaan dokumenttien määrän vähyyteen aikaikkunan sisällä. Flatline-sääntötyyppi ei siis lähetä dokumenttia hälyttimelle. Sääntö, jossa on määritetty, että kyselyssä käytetään Elasticsearchin count-ohjelmointirajapintaa ei myöskään lähetä dokumenttia hälyttimelle. Tämä johtuu siitä, että Elasticsearchin count-ohjelmointirajapintaa käytettäessä, Elasticsearch ei palauta ElastAlertille dokumentteja, vaan pelkästään suodattimiin sopivien dokumenttien määrän kyselyn aikaikkunan sisällä. (Yelp 2020e.) Näissä tapauksissa pitää hälyttimen sisällä indeksi selvittää lukemalla säännössä määritetty index-kenttä. Tästä kuitenkin seuraa se, että samaa sääntöä ei voi näissä tapauksissa käyttää monen indeksin kanssa. Hälyttimen sisällä on mahdotonta tietää mistä indeksistä hälytys on lauennut, jos säännössä on listattuna monta eri indeksiä tai indeksikuvio.

Kuvassa 16 esitetyssä hälyttimessä yritetään ensin lukea sääntöön osuneesta dokumentista sen indeksi. Jos dokumentissa ei ole index-kenttää, niin silloin indeksi haetaan

vaihtoehtoisesti säännöstä. Tämä toiminnallisuus kannattaisi tulevaisuudessa jakaa kahteen eri hälyttimeen tai ainakin vaatia säännössä manuaalinen konfiguraatioparametri, että kumpaa tapaa käytetään. Tällä hetkellä oletetaan, että säännön luoja tietää missä tilanteissa hälyttimelle lähetään dokumentti ja missä ei. Tämä toiminnallisuus voi johtaa virhetilanteisiin. Indeksin selvittämisen jälkeen hälytykseen lisätään vielä aikaleima, säännön nimi ja säännön kuvaus. Sen jälkeen lista lähetetään Elasticin Python-ohjelmointikielelle luomalla asiakasohjelmalla Elasticsearchiin.

```
from elastalert.alerts import Alerter
from elasticsearch import Elasticsearch
from elastalert.util import dt_to_ts
from elastalert.util import ts_now

class MatchIndex(Alerter):
    def alert(self, matches):
        for match in matches:

            if '_index' in match:
                alert_index = match['_index'] + "_alerts"
            else:
                alert_index = self.rule['index'] + "_alerts"

            alert_body = {'match_body': match, 'rule_name': self.rule['name']}
            if '@timestamp' not in alert_body:
                alert_body['@timestamp'] = dt_to_ts(ts_now())
            if 'description' in self.rule:
                alert_body['rule_description'] = self.rule['description']

            es = Elasticsearch([{'host': self.rule['es_host'],'port': self.rule['es_port']}])
            res = es.index(index = alert_index, doc_type = 'Alert', body = alert_body)


    def get_info(self):
        return {'type': 'Match Index'}
```

Kuva 16. MatchIndex.py-hälytin.

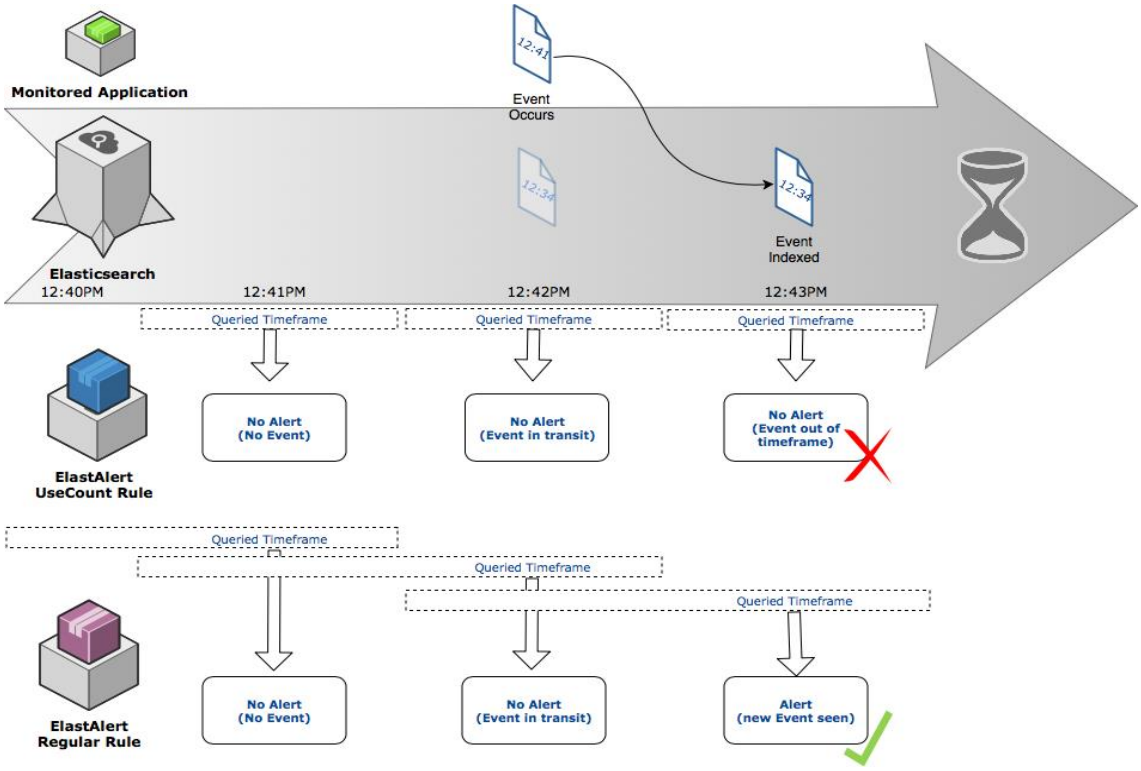
5.3 Flatline-sääntötyypin testaus

Kuvan 17 flatline-säännössä suodatetaan haettuja dokumentteja agentin tyypin perustella. Käytännössä sääntö toimii siis niin, että jos Winlogbeat-agentilta ei tule Elasticsearchiin yhtään dokumenttia kahden minuutin sisällä, niin tehdään hälytys. Hälyttimenä käytetään luvussa 5.2 kuvailtua MatchIndex.py-komentosarjaa. Säännössä määritettään myös se, että Elasticsearchista kysellessä käytetään count-ohjelmointirajapintaa.

```
es_host: 192.168.88.60
es_port: 9200
index: corp-win

name: FLATLINE - No data from winlogbeat
description: No events from winlogbeat for the past 2 minutes

type: flatline
threshold: 1
timeframe:
  minutes: 2

realert:
  minutes: 5

use_count_query: true
doc_type: _doc

query_delay:
  minutes: 1

filter:
- query:
    query_string:
      query: "agent.type: winlogbeat"

alert: "alerts.MatchIndex.MatchIndex"
```

Kuva 17. Flatline-konseptisääntö.

Count-ohjelmointirajapintaa käytettäessä on tärkeää, että säännössä asetetaan viivettä. Normaalisti ElastAlertin tekemät kyselyt Elasticsearchiin lomittuvat keskenään. Count-ohjelmointirajapintaa käytettäessä, Elasticsearch ei palauta ElastAlertille dokumentteja vaan niiden määrän. Tästä seuraa se, että ElastAlert ei voi tietää mitkä dokumentit se on aikaisemmin nähnyt. Tämän takia count-ohjelmointirajapintaa käytettäessä Elasticsearchiin lähetetyt kyselyt eivät voi lomittua. Voi siis käydä niin, että dokumentti saapuu Elasticsearchiin myöhässä ja dokumentin aikaleima on sellaisella alueella minkä ElastAlert on jo tutkinut. Tämän kaltaisessa tapauksessa, ElastAlert ei saa koskaan tietää tästä dokumentista. Asettamalla sääntöön viivettä, voidaan ongelma kuitenkin korjata. Esimerkiksi 5 minuutin viiveellä saadaan vaikutus, että lokin aikaleima saa erota 5 minuutilla hetkestä, milloin dokumentti indeksoidaan Elasticsearchiin. Viive aiheuttaa myös sen, että hälytys tulee viiveen verran myöhässä. (AUTO1 Group 2018.) Se ei kuitenkaan ole erityisen suuri ongelma muutaman minuutin viiveen kanssa. Kuvassa 18 on havainnollistettu count-ohjelmointirajapinnan toimintaa ElastAlertin kanssa.



Kuva 18. ElastAlertin toimintaa Count-ohjelmointirajapinnan kanssa havainnollistava kaavio (AUTO1 Group 2018).

Testissä sääntö toimi oletetulla tavalla. 3 minuuttia Winlogbeat-agentin pysäyttämisen jälkeen, Elasticsearch-tietokantaan lähetettiin hälytys. Kappaleessa 5.2 kuvailtu hälytin toimi myös halutusti. Indeksi johon hälytys kirjoitettiin, muodostettiin sääntötiedostossa määritetystä corp-win-indeksistä. Kuvassa 19 näkyy Elasticsearchiin kirjoitettu hälytys.

	Time	_index	rule_name
>	Apr 20, 2020 @ 21:15:39.885	corp-win_alerts	FLATLINE - No data from winlogbeat

Kuva 19. ElastAlertin tekemä hälytys Kibanan Discovery-työkalussa.

5.4 Spike-sääntötyypin testaus

Spike-sääntötyyppiä käytetään piikkien havaitsemiseen. Piikit voivat muodostua dokumenttien määrän äkillisestä kasvusta tai vähenemisestä. Kuvassa 20 on spike-sääntötyypin testausta varten luotu sääntö. Säännöllä haetaan Windows-toimialueen DC-palvelimen muodostamia lokeja, joiden tunnus on 4771. Epäonnistuneita 4771-tapahtumia voi muodostua esimerkiksi tilanteessa, jossa käyttäjän kirjautuminen Windows-

toimialueeseen liitetyllä työasemalla epäonnistuu. (Microsoft 2017.) Suuressa tai jopa keskisuuressa Windows-toimialueessa epäonnistuneita kirjautumisia tapahtuu tasaisin väliajoin. Epäonnistuneiden kirjautumisien määrän äkillinen kasvu voi johtua useasta eri syystä, kuten esimerkiksi konfiguraatiovirheestä.

Password spray -hyökkäys Kerberos-protokollalla kohdistettuna Windows-toimialueen käyttäjiin aiheuttaa myös suuren määrän 4771-tapahtumia. Password spray -hyökkäys on brute-force-hyökkäyksen variantti. Brute-force-hyökkäyksessä kohteena on yksi käyttäjä, jonka salasana yritetään arvata monta kertaa lyhyen ajan sisällä. Password spray -hyökkäyksen kohteena on monta käyttäjää. Hyökkäyksessä voidaan esimerkiksi valita kohteeksi tuhat käyttäjää, ja sen jälkeen yrittää autentikoitua jokaisella käyttäjällä, käyttäen yhtä valittua salasanaa. Password spray -hyökkäyksen suurin etu verrattuna traditionaalisempaan brute-force-hyökkäykseen on lockout-käytäntöjen kiertäminen. Käyttäjätunnukset eivät siis mene niin sanotusti lukkoon, koska kirjautumista yritettiin vain yhden kerran jokaisen käyttäjätunnuksen kanssa. Password spray -hyökkäys vaatii toimiakseen kuitenkin suuren joukon hyökättäviä käyttäjätunnuksia. Mitä enemmän hyökkääjällä on tiedossaan käyttäjätunnuksia, sitä suurempi todennäköisyys on, että password spray -hyökkäys onnistuu. (Coalfire 2019.)

Kuvan 20 säännössä vertaillaan kahta 3 minuutin kokoista aikaikkunaa keskenään. Hälytys lähetetään, jos nykyisessä aikaikkunassa on kolme kertaa enemmän dokumentteja verrattuna edelliseen aikaikkunaan. Spike-sääntötyyppi saattaa aiheuttaa vääriä hälytyksiä, esimerkiksi tilanteessa, jossa lokit tulevat takautuvasti datakatkon jälkeen. Säännössä voidaan kuitenkin asettaa lisäehto, jossa vaaditaan, että aikaisemmassa aikaikkunassa tarvitsee olla vähintään kynnysarvon verran dokumentteja.

```
es_host: 192.168.88.60
es_port: 9200
name: SPIKE - Kerberos pre-authentication failed - 4771
description: Triggered when current window is at least 3 times as high as the reference window
type: spike
index: corp-win

threshold_ref: 30
#threshold_cur: 15

timeframe:
  minutes: 3

spike_height: 3
spike_type: "up"

filter:
- query:
    query_string:
      query: "winlog.event_id: 4771"

alert: "alerts.MatchIndex.MatchIndex"
```

Kuva 20. Spike-konseptisääntö.

Voi olla myös tilanteita jossa, jokin säännöllisesti ajastettu toiminto aiheuttaa spike-säännön laukeamisen samaan kellonaikaan. Spike-sääntö, jolla hälytetään liikenteen suuresta kasvusta tietoverkossa voi laueta esimerkiksi verkon yli tapahtuvasta ajastetusta varmuuskopioinnista. Esimerkin väärää hälytystä on vaikeaa vaimentaa suoraan itse säännössä. Voidaan kuitenkin luoda enhancement-moduuli, jonka avulla hälytys pudotetaan, jos se tapahtuu tiettyyn kellonaikaan. Jotta, tätä moduulia voisi käyttää useamman eri säännön kanssa, kannattaa moduuli suunnitella niin, että hiljennettävät kellonajat määritetään sääntötiedoston sisällä.

Kuvan 21 Python-komentosarjalla simuloidaan keskisuurta Windows-toimialuetta. Kun komentosarja ajetaan DC-palvelimella se aiheuttaa noin 20-40 4771-tapahtumaa minuutissa. Tämän kaltaisella komentosarjalla ei voida kuitenkaan tarkasti simuloida todellista ympäristöä, jossa liikkuvia osia on paljon enemmän.

```
import subprocess
import time
import random
import math

users = []


while(True):
    failcount = random.randint(20,40)
    interval = 60 / failcount
    for x in range(1,math.floor(60/interval)):
        time.sleep(interval)
        f = open("NUL", "w")
        subprocess.run(
        ["kerbrute.exe", "passwordspray", "-d", "corp.local", "user.txt", "FAILPASS"], stdout=f)
        f.close()
```

Kuva 21. Python-komentosarja taustakohinan luomiselle.

Toimialueeseen liitetyllä työasemalla, hyökkääjä voi saada tietoonsa toimialueen käyttäjätunnuksia esimerkiksi kuvan 22 komennolla. Tämän jälkeen kuvitteellinen hyökkääjä voi suorittaa password spray -hyökkäyksen löytämiään käyttäjätunnuksia vastaan. Yksi tapa suorittaa Windows-toimialueella password spray -hyökkäys on käyttää Kerbrute-työkalua. Kerbrute-työkalulla voi suorittaa myös muita Kerberos-protokollaan kohdistuvia hyökkäyksiä. Työkalulla pystyy esimerkiksi suorittamaan brute-force-hyökkäyksen yhtä käyttäjää vastaan. Kerbrute-hyökkäystyökalu on helposti ladattavissa GitHubista. (Flathers, R. 2019.)

```
C:\Users\jh3452>net user /domain
The request will be processed at a domain controller for domain corp.local.


User accounts for \\DC1.corp.local

-------------------------------------------------------------------------------
0backup                  0ur6621                  ad2854
ae1576                   al8290                   am6630
as1177                   aw9192                   ay3235
az6812                   bg3242                   bh9595
bp8389                   br9802                   bz5417
cg2796                   ck2941                   cu2263
```

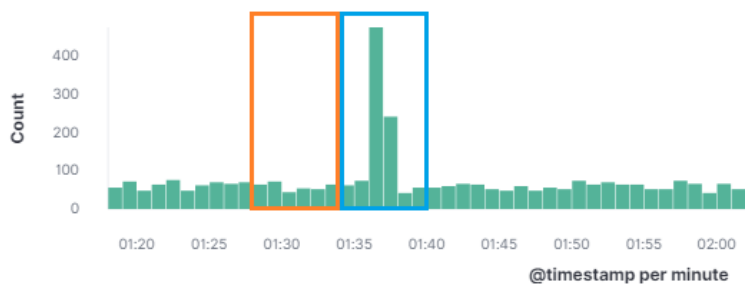
Kuva 22. net user /domain -komennon tuloste.

Kuvassa 23 password spray -hyökkäys toteutetaan Kerbrute-työkalulla. Työkalun tulosteesta saa selville, että testattu salasana toimii kp8121-käyttäjällä. Varsinainen hyökkäys 106 käyttäjätunnusta vastaan ei kestänyt kuin 0.662 sekuntia.

```
C:\Users\jh3452>kerbrute_windows_amd64.exe passwordspray -d corp.local DomainUsersParsed.txt Password2

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 04/12/20 - Ronnie Flathers @ropnop

2020/04/12 01:36:14 >  Using KDC(s):
2020/04/12 01:36:14 >   DC1.corp.local:88
2020/04/12 01:36:14 >  [+] VALID LOGIN:  kp8121@corp.local:Password2
2020/04/12 01:36:14 >  Done! Tested 106 logins (1 successes) in 0.662 seconds
```

Kuva 23. Onnistunut Password Spray -hyökkäys Kerbrute-hyökkäystyökalulla.

Kuvassa 24 oranssilla ja sinisellä laatikolla havainnollistetaan ElastAlertin spike-säännössä vertailemia aikaikkunoita. Kuvasta on helppo nähdä jo pelkästään silmämääräisesti, että sinisellä kuvattu nykyinen-aikaikkuna sisältää yli kolme kertaa enemmän tapahtumia kuin aikaisempi oranssi aikaikkuna. Password spray -hyökkäyksen luomat 4771-tapahtumat aiheuttivat kuvassa 20 havainnollistetun säännön laukeamisen.



Kuva 24. Password Spray -hyökkäyksen aiheuttama piikki 4771-tapahtumissa.

5.5 Query_Key-kenttä

ElastAlertin dokumentaatio ei ole aina ajan tasalla. Dokumentaatiossa ei esimerkiksi mainita, että yhdessä säännössä voidaan listata useampia eri indeksejä. Säännössä testattavia indeksejä voi kuitenkin listata index-kenttään pilkulla erotettuina. Tämän kaltaisissa säännöissä ilmenee kuitenkin ongelma tilanteessa, jossa eri indeksien tulokset halutaan pitää erillään. Jos säännössä ei määritellä query_key-arvoa niin sääntö ei erottele keskenään eri indeksistä löytämiään tuloksia. Kuvan 25 sääntö hälyttää kun, yhdestä indeksistä löytyy yli kolme dokumenttia minuutin kokoisen aikaikkunan sisältä. Jos query_key-kentällä ei eroteltaisi dokumentteja siitä löytyneen indeksikentän perusteella, niin hälytys laukeaisi myös muissa tilanteissa. Hälytys laukeaisi esimerkiksi tilanteessa, jossa toisesta indeksistä löytyisi minuutin sisällä kaksi dokumenttia ja toisesta indeksistä löytyisi myös kaksi dokumenttia. Sääntö ei erottelisi tuloksia millään tavalla vaan hälyttäisi kolmannen dokumentin nähdessään. Query_key-kenttää ei voida käyttää tällä tavalla flatline-säännöissä, eikä säännöissä, joissa käytetään Elasticsearchin count-ohjelmointirajapintaa. Query_key-kenttää voidaan kuitenkin käyttää esimerkiksi spike- ja frequency-sääntötyypeissä.

Query_key-kentän toiminta vaihtelee hieman eri sääntötyyppien välillä. Omaa sääntötyyppiä luodessa, voidaan sen toiminta suunnitella sillä tavalla, että samassa säännössä on mahdollista listata useita eri indeksejä. Säännöstön koko saattaa muussa tapauksessa paisua suureksi, jos testattavia indeksejä on paljon. Monen eri säännön tekeminen ja hallitseminen on työläämpää kuin yhden säännön muokkaaminen.

```
es_host: 192.168.88.60
es_port: 9200
name: Querykey test rule
description: Test alerting on 2 indexes with 1 rule
type: frequency

index: "windc_filebeat,winpc_filebeat"

num_events: 3
query_key: "_index"

timeframe:
    minutes: 1
filter: []

alert: "alerts.MatchIndex.MatchIndex"
```

Kuva 25. Frequency-sääntö query_key-kentän toiminnan testaamiselle.

6 POHDINTA

Opinnäytetyön tavoitteena oli tutkia ElastAlert-työkalun käyttömahdollisuuksia tietoturvapoikkeamien havainnointiin. On hyvin vähän lähteitä, joissa ElastAlertin toimintaa analysoidaan tätä käyttötarkoitusta varten. Julkisissa projekteissa ja artikkeleissa ElastAlerttia on käytetty tietoturvapoikkeamien havainnointiin lähinnä pelkästään any-sääntötyypin kanssa. Säännöllä voidaan esimerkiksi luoda hälytys tietystä merkkijonosta lokitapahtuman sisällä. Tämän kaltaiset säännöt toimivat lähes samalla tavalla kuin traditionaaliset tunkeilijan havaitsemisjärjestelmät. ElastAlertissa on kuitenkin paljon sellaisia toiminnallisuuksia, joita traditionaalisemmissa järjestelmissä ei ole. Yksi esimerkki tämän kaltaisesta toiminnallisuudesta on, opinnäytetyön testausvaiheessa havainnollistettu spike-sääntötyyppi.

ElastAlert on suunniteltu hyvin yksinkertaiseksi, mutta kuitenkin erittäin joustavaksi työkaluksi. Erilaisten valmiiden ja varsinkin itserakennettujen moduulien käyttö tekeekin ElastAlertista houkuttavan työkalun. Python-ohjelmointikielen valinta oli ElastAlertin kehittäjiltä onnistunut päätös. Python on maailman kolmanneksi suosituin ohjelmointikieli (TIOBE 2020). Pythonille on myös todella suuri määrä valmiita kirjastoja, joiden käyttö omien hälyttimien, enhancement-moduulien tai sääntötyyppien sisällä on helppoa. Omien moduulien tekeminen on myös suunniteltu mahdollisimman yksinkertaiseksi. Opinnäytetyössä esitetyt säännöt olivat yksinkertaisia, mutta rakentamalla omia sääntötyyppejä ja enhancement-moduuleja on mahdollista tehdä paljon monimutkaisempia sääntöjä. Itse ElastAlert-työkalunkin muokkaaminen on mahdollista avoimen lähdekoodin takia. Ensisijaisesti kuitenkin uudet toiminnallisuudet kannattaa yrittää toteuttaa itsetehdyillä moduuleilla.

ElastAlertin onnistunut implementaatio poikkeamien havainnointia varten vaatii organisaatiolta paljon tietoturvaosaamista sekä kokemusta tietoturvapoikkeamien havainnoinnista. ElastAlert on niin sanotusti tyhjä taulu, johon koko säännöstö pitää kehittää itse. Voi olla, että tulevaisuudessa julkaistaan enemmän ElastAlertilla tehtyjä projekteja ja sille kehitettyjä säännöstöjä. Tämä ei kuitenkaan hyvin kuvaa ole tämän hetkistä tilannetta. Tietoturvapoikkeamien havainnointiin käytettävä säännöstö vaati myös jatkuvaa ylläpitoa. Hyökkääjien käyttämät metodit ja teknologiat muuttuvat ja kehittyvät joka päivä. Säännöstön ylläpidon tärkeys on verrattavissa virustorjuntaohjelmistojen ylläpidon tärkeyteen. Esimerkiksi virustorjuntaohjelmisto vuodelta 2010 havaitsee sen jälkeen

kehitettyjä haittaohjelmia erittäin heikosti. Sääntöjen tekemisen suurimmat rajoittavat tekijät ovat oma mielikuvitus sekä Elasticsearchin turhan suuri rasittaminen. On helppoa ideoida sääntöjä, jotka eivät kuitenkaan ole realistisia niiden hakemien dokumenttimäärien vuoksi.

Opinnäytetyön testausvaiheessa suoritettu testaus ei ole vielä täysin riittävää. Testit suoritettiin ympäristössä, jossa havainnoitavan ympäristön tapahtumalokit olivat täysin testaajan hallinnassa. Todellisessa ympäristössä on paljon enemmän liikkuvia osia. Piikkien ja muiden vastaavien tilastollisien poikkeamien havainnointi voi osoittautua monimutkaisemmaksi. Ei ole esimerkiksi täysin selkeää, miten spike-sääntötyyppi käyttäytyy erilaisten datakatkojen kanssa ja miten normaali työpäivärytmi vaikuttaa erilaisiin sääntötyyppeihin. Esimerkiksi tietojärjestelmissä tapahtuvat epäonnistuneet kirjatumiset lisääntyvät suuresti aamulla, kun työntekijät saapuvat työpisteilleen.

Jatkokehityksessä, ElastAlertille pitää luoda toimiva säännöstö ja säännöstön toimintaa tukevat moduulit. On myös tärkeää, että ElastAlertin toimintaa päästään testaamaan oikeassa ympäristössä. Seuraavassa testausvaiheessa on mahdollista, että ElastAlertin toiminnasta löytyy sellaisia puutteita, että varsinaista työkalua joudutaan muokkaamaan.

Opinnäytetyön suunnitteluvaiheessa asetetut tavoitteet saavutettiin. Tutkimus- ja testausvaiheiden avulla ElastAlertin eri toiminnallisuuksista ja ominaisuuksista saatiin kattava kuva. Opinnäytetyössä kootun tiedon avulla pystyi tekemään päätöksen siitä, että onko ElastAlert sopiva työkalu sitä harkitsevalle organisaatiolle poikkeamien havainnointiin.

LÄHTEET

0x29A tech blog 2019. [Elastalert] Enable alert only in specific hour range. Viitattu 27.4.2020 https://marchal.tech/blog/2019/08/27/elastalert-enable-alert-only-in-specific-hour-range/.

AUTO1 Group 2018. ElastAlert Tips & Tricks. Viitattu 28.4.2020 https://auto1.tech/elastalert/.

Bejtlich, R. 2013. The Practice Of Network Security Monitoring. ISBN 9781593275099.

Bitsensor 2020. ElastAlert Kibana Plugin. Viitattu 27.4.2020 https://github.com/bitsensor/elastalert-kibana-plugin.

Coalfire 2019. “Password Spraying”—What to Do and How to Avoid It. Viitattu 28.4.2020 https://www.coalfire.com/The-Coalfire-Blog/March-2019/Password-Spraying-What-to-Do-and-How-to-Avoid-It.

Couchbase N.d. Comparing document databases and relational databases. Viitattu 24.4.2020 https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/compare-docs-vs-relational.html.

Cybereason 2019. Operation Soft Cell: A Worldwide campaign against telecommunications providers. Viitattu 09.06.2020 https://www.cybereason.com/blog/operation-soft-cell-a-worldwide-campaign-against-telecommunications-providers

Elastic 2017. A Practical Introduction to Elasticsearch. Viitattu 25.4.2020 https://www.elastic.co/blog/a-practical-introduction-to-elasticsearch.

Elastic N.d.a. Stories from users like you. Viitattu 24.4.2020 https://www.elastic.co/customers/.

Elastic N.d.b. Data in: documents and indices. Viitattu 24.4.2020 https://www.elastic.co/guide/en/elasticsearch/reference/7.6/documents-indices.html.

Elastic N.d.c. Information out: search and analyze. Viitattu 24.4.2020 https://www.elastic.co/guide/en/elasticsearch/reference/7.6/search-analyze.html.

Elastic N.d.d. Scalability and resilience: clusters, nodes, and shards. Viitattu 24.4.2020 https://www.elastic.co/guide/en/elasticsearch/reference/7.6/scalability.html.

Elastic N.d.e. Logstash Introduction. Viitattu 24.4.2020 https://www.elastic.co/guide/en/logstash/current/introduction.html.

Elastic N.d.f. Kibana — your window into the Elastic Stack. Viitattu 24.4.2020 https://www.elastic.co/guide/en/kibana/current/introduction.html.

Elastic N.d.g. Beats overview. Viitattu 24.4.2020 https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html.

Elastic N.d.h. Alerting. Viitattu 25.4.2020 https://www.elastic.co/what-is/elasticsearch-alerting.

Fellinge J. 2003. Snort Rules. Viitattu 08.06.2020 https://www.itprotoday.com/security/snort-rules

Flathers, R. 2019. Kerbrute. Viitattu 28.4.2020 https://github.com/ropnop/kerbrute.

HELK 2020. The Hunting ELK. Viitattu 25.4.2020 https://github.com/Cyb3rWard0g/HELK.

Lockheed Martin N.d. Cyber Kill Chain. Viitattu 08.06.2020 https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

Microsoft 2017. 4771(F): Kerberos pre-authentication failed. Viitattu 28.4.2020 https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4771.

Mitre 2020. ATT&CK. Viitattu 10.06.2020 https://attack.mitre.org/

Prakasha S. 2016. The Growing Popularity of the Snort Network IDS. Viitattu 08.06.2020 https://opensourceforu.com/2016/09/growing-popularity-snort-network-ids/

TIOBE 2020. TIOBE Index for April 2020. Viitattu 25.4.2020 https://www.tiobe.com/tiobe-index/.

Walkowski, D. 2019. What is the CIA Triad?. Viitattu 15.5.2020 https://www.f5.com/labs/articles/education/what-is-the-cia-triad

Yelp 2019. Fast Facts. Viitattu 25.4.2020 https://www.yelp-press.com/company/fast-facts/default.aspx.

Yelp 2020a. ElastAlert - Easy & Flexible Alerting With Elasticsearch. Viitattu 25.4.2020 https://elastalert.readthedocs.io/en/latest/elastalert.html.

Yelp 2020b. ElastAlert Insights. Viitattu 25.4.2020 https://github.com/Yelp/elastalert/pulse.

Yelp 2020c. Python Package Index: ElastAlert. Viitattu 25.4.2020 https://pypi.org/project/elastalert/#history.

Yelp 2020d. Running ElastAlert for the First Time. Viitattu 30.4.2020 https://elastalert.readthedocs.io/en/latest/running_elastalert.html.

Yelp 2020e. Rule Types and Configuration Options. Viitattu 25.4.2020 https://elastalert.readthedocs.io/en/latest/ruletypes.html.

Yelp 2020f. Writing Filters For Rules. Viitattu 25.4.2020 https://elastalert.readthedocs.io/en/latest/recipes/writing_filters.html.

Yelp 2020g. Adding a New Rule Type. Viitattu 25.4.2020 https://elastalert.readthedocs.io/en/latest/recipes/adding_rules.html.

Yelp 2020h. Adding a New Alerter. Viitattu 25.4.2020 https://elastalert.readthedocs.io/en/latest/recipes/adding_alerts.html.

Yelp 2020i. Enhancements. Viitattu 27.4.2020 https://elastalert.readthedocs.io/en/latest/recipes/adding_enhancements.html#.

Yelp 2020j. Rules Loaders. Viitattu 30.4.2020 https://elastalert.readthedocs.io/en/latest/recipes/adding_loaders.html.